COMPLEXITY ANALYSIS OF THE COST-TABLE APPROACH TO THE DESIGN OF MULTIPLE-VALUED LOGIC CIRCUITS*

by

Kriss A. Schueller
Department of Mathematics and Computer Science
Youngstown State University
Youngstown, OH 44555-3134

and

Jon T. Butler
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93943-5121

October 8, 1995

*A preliminary version of this manuscript appeared in the *Proceedings of the 28th Annual Allerton Conference on Communication, Control, and Computing,* October 1990.

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **08 OCT 1995**	2. REPORT TYPE	3. DATES COVERED
4. TITLE AND SUBTITLE **Complexity Analysis of the Cost-Table Approach to the Design of Multiple-Valued Logic Circuits**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Department of Electrical and Computer Engineering,Monterey,CA,93943**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited.**		
13. SUPPLEMENTARY NOTES		

14. ABSTRACT

We analyze the computational complexity of the cost-table approach to designing multiple alued logic circuits that is applicable to I L, CCD?s, current-mode CMOS, and RTD?s. We s 2 how that this approach is NP-complete. An efficient algorithm is shown for finding the exact I minimal realization of a given function by a given cost-table.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**18**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

ABSTRACT

We analyze the computational complexity of the cost-table approach to designing multiple-valued logic circuits that is applicable to I^2L, CCD's, current-mode CMOS, and RTD's. We show that this approach is NP-complete. An efficient algorithm is shown for finding the exact minimal realization of a given function by a given cost-table.

Index terms: *computational complexity*, *cost-table*, *cost function*, *logic design*, *minimization*, *multiple-valued logic*, *NP-complete*, *synthesis*

I. INTRODUCTION

The first demonstration that a logic synthesis problem is NP complete occurred as the result of two insights. To find the minimal sum-of-products expression for a logic function, one can produce the set S of all prime implicants and then use a minimal subset of S to cover all minterms of the function. The latter step is a specific case of the set covering problem. Because it is specific case, it is possible that it is not as complex as the general set covering problem. However, Gimpel [2] showed that this is not true. He showed that any instance of the set covering problem occurs as an instance of the sum-of-products problem. Subsequently, Karp [3] proved that the set covering problem is NP-complete; thus, proving that extracting a minimal sum-of-products expression is NP-complete.[1] While complexity questions have frequently occurred in *multiple-valued* logic (e.g. [1,7]), there has been no classification of the synthesis of multiple-valued functions complexity classes, e.g. NP-completeness.

The need for design techniques for multiple-valued CCD circuits, [5], inspired interest in the cost-table approach, e.g. [1, 6, 7]. In the cost-table approach, a given function is realized by selecting functions from a table and combining them. Associated with each chosen function is a cost, which can represent chip area, power dissipation, speed, etc. The cost of a realization is the sum of the costs of the component functions plus the cost of combining them. Usually, there is more than one way to realize a given function, and the goal of the design is to find a realization of lowest cost. This is called the *Cost-table Realization* problem. The question posed and answered in this paper is "How the does the time to solve the cost-table realization problem depend on the size of the cost-table?". We show that this problem is NP-complete.

II. BACKGROUND AND NOTATION

A function $f(X)$ is a mapping $f : D^n \rightarrow R$, where $D = \{0, 1, \ldots, d-1\}$ and

[1]Keutzer and Richards [4] point out that there has been misunderstanding in certain papers on the complexity of the sum-of-products extraction problem. That is, the problem of finding a sum-of-products expression with no more than some given number of terms is NP-complete if the function is expressed as a truth table, but co-NP hard if the function is expressed as a sum-of-products expression.

$R = \{0, 1, \ldots, r-1\}$.

When $n = 1$, it is convenient to represent $f(X)$ in the form $<f(0), f(1), \cdots, f(d-1)>$. For example, if $d = r = 4$, then $f(X) = <3,2,1,0>$ is the four-variable *complement* function. The set of all r-valued functions of n d-valued variables is $U^r_{d,n}$. Let $c(f)$, the *cost function*, be a mapping $c : U^r_{d,n} \rightarrow \boldsymbol{R}^{0+}$, where $\boldsymbol{R}^{0+}$ is the set of nonnegative real numbers. For example, the cost function $c(f)$ introduced by Kerkhoff and Robroek [6] for the design of 4-valued CCD logic circuits correlates closely with the chip area occupied by the most compact implementation of f.

Given a function $f(X)$ to be realized using a cost-table, we seek a representation of the form $f(X) = f_1(X) \boldsymbol{+} f_2(X) \boldsymbol{+} \cdots \boldsymbol{+} f_m(X)$, where $\boldsymbol{+}$ is ordinary addition with logic values viewed as integers. For example, if $f_1(X) = <0, 1, 2, 3>$ and $f_2(X) = <3, 2, 1, 0>$, then $f_1(X) \boldsymbol{+} f_2(X) = <3, 3, 3, 3>$. In our analysis, it is convenient to assume that the sum of two logic values does not exceed the highest logic value, $r-1$. Thus, $\boldsymbol{+}$ can be implemented as the sum mod r or as truncated sum, for example. The latter is more common in practice, since it is easily implemented, e.g. in CCD or current-mode logic. The effect of this assumption is not to restrict the operations possible, but the synthesis technique. For example, $f_1 \boldsymbol{+} f_1$ is not a realization of the synthesis technique because two components sum to a value greater than $r - 1$. Let σ be the cost of realizing the sum of two functions. The cost of the realization $f = f_1 \boldsymbol{+} f_2 \boldsymbol{+} \cdots \boldsymbol{+} f_m$ is

$$c(f_1) + c(f_2) + \cdots + c(f_m) + (m-1)\sigma,$$

where σ is the cost of combining two cost-table functions.

A *basis function* f has the property that $f(A)$ is 1 for exactly one assignment A of values to X and is 0 for all other assignments. Let BT be the set of all basis functions plus **0**, the function that is 0 for all assignments of values to the variables (e.g., $<0,0,0,0>$). BT is called the *basis cost-table*. F is a *cost-table* if and only if $BT \subseteq F \subseteq U^r_{d,n}$. Note that all functions in BT are needed in F. Indeed, if the function f to be realized has the property $f \in BT$, then f cannot be realized, unless $f \in F$. Of all the ways to realize a given function f using cost-table F, one realization, $f = f_1 + f_2 + \cdots + f_m$, where $f_i \in F$, has a cost that is lower than or equal to the cost of all other realizations of f using F. Denote realization $f = f_1 + f_2 + \cdots + f_m$ as a *minimal cost realization of* f. Note that, there may be more than one such realizations. Its cost, $c(f_1) + c(f_2) + \cdots + c(f_m) + (m-1)\sigma$, is the *cost of realizing* $f \in U^r_{d,n}$ *using cost-table F*, and will be denoted as $c_F(f)$. Thus, whenever we seek the cost of realizing a given function f using a given cost-table F, we assume that, of all the ways to realize a function f using cost-table F, we choose the lowest cost realization. Formally,

$$c_F(f) = \min_{\substack{f_1, f_2, \ldots, f_m \in F \\ f = f_1 + f_2 + \cdots + f_m}} \{c(f_1) + c(f_2) + \cdots + c(f_m) + (m-1)\sigma\},$$

The *total cost*, $T(F)$, of cost-table F is

$$T(F) = \sum_{f \in U^r_{d,n}} c_F(f).$$

F is a *minimal* cost-table if $T(F) \le T(F')$, for all F', such that $|F| = |F'|$, where $|F|$ is the cardinality of F. The term "minimal" describes the cost over *all* realizations of a cost-table.

The *(Minimal) Cost-table Realization*, (MCR) CR, problem is:

Given a (minimal) cost-table F, a function f, and a cost function c, find a minimal cost realization $f = f_1 + f_2 + \cdots + f_m$, where $f_i \in F$.

The *(Minimal) Cost-table Decision*, (MCD) CD, problem is:

Given a (minimal) cost-table F, a function f, a cost function c, and a target cost P, does there exist a realization $f = f_1 + f_2 + \cdots + f_m$, such that $c(f_1 + f_2 + \cdots + f_m) \le P$, where $f_i \in F$?

Let (MCD(F, f, c, P)) CD(F, f, c, P) denote an instance of this problem. (MCD(F, f, c, P)) CD(F, f, c, P) is said to be satisfied if and only if such a realization exists. The *size* K of an instance of (MCD(F, f, c, P)) CD(F, f, c, P) is $d^n\,|F|$. K accounts for both the function size, as well as the cost-table size. Since the MCD(F, f, c, P) is a special case of the CD(F, f, c, P), there is the possibility that it is not as complex. We show, however, that this is not the case.

III. COMPLEXITY OF THE COST-TABLE REALIZATION PROBLEM

The main results are presented in two theorems.

Theorem 1: The Cost-table Decision problem is NP-complete.

Theorem 2: The Minimal Cost-table Decision problem is NP-complete.

We proceed by first showing that these two problems are within NP; that is, we show in, Lemma 1, that there exists a non-deterministic Turing Machine that calculates each problem in time polynomial in the size of the problem.

Next, in Lemma 2, we show that there is a polynomial time transformation of the Knapsack problem to the (Minimal) Cost-table Decision Problem, where the former is satisfied iff the latter is satisfied. Since the Knapsack problem is known to be NP-complete, this shows that the (Minimal) Cost-table Decision problem is NP-complete.

Consider the solution of (MCD(F, f, c, P)) CD(F, f, c, P) by a non-deterministic algorithm that scans F, choosing as many as $r - 1$ copies of each function for each of the d^n possible assignments of values to the variables. This can be done in no more than $\mathbf{O}((r-1)\, d^n\, |F|)$ time. This algorithm can check whether the chosen function is a realization of f in $\mathbf{O}(d^n)$ time. Also, it can check whether the cost is less than or equal to P in $\mathbf{O}((r-1)\, |F|)$ time. Since the *size* of an instance of this problem is $K = d^n\, |F|$, this proves the following.

Lemma 1: There exists a non-deterministic algorithm that solves (MCD(F, f, c, P)) CD(F, f, c, P) in time that is polynomial in its size.

The *Knapsack Decision* problem can be stated as follows:

> Given a set Q of objects, a size function $s: Q \rightarrow \mathbf{Z}^+$, a value function $v: Q \rightarrow \mathbf{Z}^+$, a size S, and a value V, is there a subset $Q' \subseteq Q$ such that $\sum_{u \in Q'} v(u) \geq V$ and $\sum_{u \in Q'} s(u) \leq S$, where $\mathbf{Z}^+$ is the set of positive integers?

Let $\mathrm{KD}(Q, s, v, S, V)$ be an instance of the Knapsack Decision problem. $\mathrm{KD}(Q, s, v, S, V)$ is said to be satisfied if and only if such a subset Q' exists. The *size* of an instance of this problem is $|Q|$.

Definition: Let Φ be a transformation from any instance of the Knapsack Decision problem to an instance of the (Minimal) Cost-table Decision problem

$$\Phi(\mathrm{KD}(Q, s, v, S, V)) = (\mathrm{MCD}(F, f, c, P)) \ \ \mathrm{CD}(F, f, c, P),$$

with F, f, c, and P defined as follows:

1) The cost-table F consists of r-valued functions on one d-valued variable, where $r = S+1$ and $d = |Q|+1$. Besides the $d+1$ functions in BT, there are $d-1$ non-basis functions f_1, $f_2, \ldots, f_{d-1}$, where f_i corresponds to u_i, the ith element in Q. Specifically, $f_i(0) = s(u_i)$, $f_i(i) = 1$, and $f_i(j) = 0$, for $1 \leq j \leq d-1$, $j \neq i$. We have

$$\begin{array}{lcl} f_1 & = & <s(u_1), 1, 0, 0, \cdots, 0> \\ f_2 & = & <s(u_2), 0, 1, 0, \cdots, 0> \\ \vdots & & \qquad\qquad \vdots \\ f_{d-1} & = & <s(u_{d-1}), 0, 0, 0, \cdots, 1>. \end{array}$$

2) Function f has the form

$$f = <S, 1, 1, 1, \cdots, 1>.$$

Since $f(i) = 1$ for $1 \le i \le d-1$, each f_i can be used at most once in the realization of f. This corresponds to the restriction that each element $u_i \in S$ is used at most once in the Knapsack Decision problem. Also, since $f(0) = S$, the sum $\sum f_i(0)$ over the f_i's used in a realization of f (i.e. $s(u_i)$) must be less than or equal to S.

3) Let $c(f_i) = s(u_i)$, for $1 \le i \le d-1$. Let the cost of functions in BT be defined as follows.

$$c(b_j) = \begin{cases} 0 & \text{if } j = 0 \\ v(u_j) & \text{otherwise} \end{cases},$$

where $b_j(j) = 1$ and $b_j(i) = 0$ for $i \ne j$. That is, the cost of $<1,0, \cdots ,0>$ is 0, while the cost of all other basis functions is the value of some object in Q. The cost of the constant function $<0,0, \cdots ,0>$ is 0. Let the cost, σ, of combining two functions be 1.

If Φ is a transformation to CD(F, f, c, P), we allow any specification of the cost of a function g, such that $g \notin F$. If Φ is a transformation to MCD(F, f, c, P), we make the additional specification that, for $g \notin F$, $c(g) = \infty$. In this way, F is a minimal cost-table; i.e. no interchange of functions outside F with functions inside F that preserves the size of the cost-table yields a total cost lower than $T(F)$.

4) P is defined by

$$P = \sum_{u_i \in Q} v(u_i) - V + (S + d - 2). \tag{1}$$

Example: Consider a knapsack defined as follows. Let $Q = \{u_1, u_2, u_3\}$, and let $s(u_i)$ and $v(u_i)$ be specified as follows.

	$s(u_i)$	$v(u_i)$
u_1	3	4
u_2	2	3
u_3	2	2

Table I: Sizes and values of elements of the knapsack.

Let $S = 5$ and $V = 6$.

Of the 8 ways to choose subsets of Q, there are two that satisfy $KD(Q, s, v, S, V)$,

$Q_1 = \{u_1, u_2\}$	$\sum_{u \in Q_1} v(u) = 7 \geq V = 6$ $\sum_{u \in Q_1} s(u) = 5 \leq S = 5$
$Q_2 = \{u_1, u_3\}$	$\sum_{u \in Q_2} v(u) = 6 \geq V = 6$ $\sum_{u \in Q_2} s(u) = 5 \leq S = 5$

Table II: The two solutions to the Knapsack Decision problem.

Applying the transformation yields a cost-table where $r = 6$ and $d = 4$ with functions

Function	Cost	
<0,0,0,0>	0	0
<1,0,0,0>	0	0
<0,1,0,0>	4	$v(u_1)$
<0,0,1,0>	3	$v(u_2)$
<0,0,0,1>	2	$v(u_3)$
<3,1,0,0>	3	$s(u_1)$
<3,0,1,0>	2	$s(u_2)$
<2,0,0,1>	2	$s(u_3)$

Table III: Cost-table as transformed from the Knapsack Decision problem.

The function to be synthesized is f = <5,1,1,1>, and P = 10. The instance of the cost-table decision problem, $CD(F, f, c, P)$ so formed, is satisfied by exactly two realizations of f, as follows.

Function	Cost	Function	Cost
<3,1,0,0>	3	<3,1,0,0>	3
<2,0,1,0>	2	<2,0,0,1>	2
<0,0,0,1>	2	<0,0,1,0>	3
Additions	2	Additions	2
Total	9	Total	10

Table IV: Two solutions to the Cost-table Decision problem.

These two realizations match left to right with $\{u_1, u_2\}$ and $\{u_1, u_3\}$, the subsets satisfying

KD(Q, s, v, S, V). Note that, of the two realizations of <5,1,1,1,1>, one is uniquely minimal, that given in the left hand column above.

We can make the following general statement.

Lemma 2: Φ is a polynomial time transformation of the Knapsack Decision problem to the (Minimal) Cost-table Decision problem, such that KD(Q, s, v, S, V) is satisfied if and only if (MCD(F, f, c, P)) CD(F, f, c, P) = Φ(KD(Q, s, v, S, V)) is satisfied.

Proof: The proof is divided into three parts. First, it is shown that Φ takes polynomial time. Then, it is shown that, if KD(Q, s, v, S, V) is satisfied, then Φ(KD(Q, s, v, S, V)) is satisfied (only if). Finally, it is shown that, if Φ(KD(Q, s, v, S, V)) is satisfied then, KD(Q, s, v, S, V) is satisfied (if).

To form the cost-table $F \subseteq U^r_{d,1}$, Φ generates $d-1 = |Q|$ non-basis functions, d basis functions, and the constant function <0,0, $\cdots$,0>. Each function can be described by a truth table with $d = |Q| + 1$ entries. An entry in the truth table can be made in constant time. Thus, the total time needed to generate F is $\mathbf{O}(|Q|^2)$. A cost is then assigned to each function requiring constant time per function. Since $s(u_i)$ can be computed in constant time, the target function f can be formed in $\mathbf{O}(|Q|)$ time. Finally, P requires the summation of all $v(u_i)$, which also takes $\mathbf{O}(|Q|)$ time. Since each step takes at most polynomial time, the entire transformation takes polynomial time.

As preparation for the next two parts, consider

$$g_i = \begin{cases} f_i & \text{if } u_i \in Q' \text{ and } 1 \le i \le d-1 \\ b_i & \text{if } u_i \notin Q' \text{ and } 1 \le i \le d-1 \\ b_0 & \text{if } d \le i \le m \end{cases} ,$$

where Q' is the subset of Q that satisfies the Knapsack Decision problem and $m = S - S' + |Q|$, for $S' = \sum_{u_i \in Q'} s(u_i)$. We now show that $g_1 + g_2 + \cdots + g_m = f$.

Consider $g_1 + g_2 + \cdots + g_m$, when the variable value is 0.

$$\sum_{i=1}^{m} g_i(0) = \sum_{u_i \in Q'} f_i(0) + \sum_{u_i \notin Q'} b_i(0) + \sum_{i=d}^{m} b_0(0)$$

$$= \sum_{u_i \in Q'} s(u_i) + 0 + (m-d+1) = S' + 0 + (S-S') = f(0).$$

When the variable value is not 0, $g_1 + g_2 + \cdots + g_m$ is evaluated as follows. By the definition of f_i and b_i, $g_i(j) = 0$ if $i \neq j$ and $1 \leq j$. Therefore, $\sum_{i=1}^{m} g_i(j) = 1 = f(j)$, for $1 \leq j \leq d-1$. This proves that $g_1 + g_2 + \cdots + g_m = f$.

The cost of realization $f = g_1 + g_2 + \cdots + g_m$ is

$$\sum_{u_i \in Q'} s(u_i) + \sum_{u_i \notin Q'} v(u_i) + 0 + (m-1)$$

or

$$S' + \sum_{u_i \in Q} v(u_i) - V' + (S - S' + |Q| - 1),$$

where $V' = \sum_{u_i \in Q'} v(u_i)$. From (1), the cost of this realization is $P - V' + V$.

(only if) Assume $\mathrm{KD}(Q, s, v, S, V)$ is satisfied by Q'. The *size* of this collection is $S' = \sum_{u_i \in Q'} s(u_i)$, and the *value* is V'. Since Q' satisfies $\mathrm{KD}(Q, s, v, S, V)$, $S' \leq S$ and $V' \geq V$. Now consider $c_F(f)$, the minimal cost realization of f in cost-table F. Because the cost of the realization $g_1 + g_2 + \cdots + g_m$ is an upper bound on the minimal cost realization, $c_F(f) \leq P - V' + V$. Since $V' \geq V$, $c_F(f) \leq P$. If F is a minimal cost-table, then $\mathrm{MCD}(F, f, c, P)$ is satisfied. Else, $\mathrm{CD}(F, f, c, P)$ is satisfied.

(if) Assume $\Phi(\mathrm{KD}(Q, s, v, S, V)) = (\mathrm{MCD}(F, f, c, P))\ \mathrm{CD}(F, f, c, P)$ is satisfied by the realization $f = h_1 + h_2 + \cdots + h_l$, where $h_i \in F$. Then, $\sum_{i=1}^{l} c(h_i) + (l-1) \leq P$. We show that the Knapsack Decision problem is satisfied for

$$Q' \;=\; \{u_i \mid h_i \notin BT\}.$$

To calculate the "size" of the solution, consider the function evaluated at 0; that is, $\sum_{i=1}^{l} h_i(0) = f(0)$. We can write

$$\sum_{u_i \in Q'} h_i(0) + \sum_{u_i \notin Q'} h_i(0) = S,$$

where the functions in the right sum are in BT, while those in the left sum are not. Since $h_i(0) \geq 0$, the right sum in the above equation is nonnegative. Therefore, $\sum_{u_i \in Q'} h_i(0) \leq S$ and thus,

$$\sum_{u_i \in Q'} s(u_i) \leq S.$$

To calculate the "value" of the solution, consider the cost of the realization $f = h_1 + h_2 + \cdots + h_l$. Because this is a solution to (MCD(F, f, c, P)) CD(F, f, c, P),

$$\sum_{i=1}^{l} c(h_i) + (l-1) \leq P.$$

Inserting the definitions of P and $c(h_i)$ into this equation yields,

$$\sum_{u_i \in Q'} s(u_i) + \sum_{u_i \notin Q'} v(u_i) + l - 1 \leq \sum_{u_i \in Q} v(u_i) - V + (S + d - 2).$$

Rearranging, yields

$$V + \left[l - [(d-1) + S - \sum_{u_i \in Q'} s(u_i)] \right] \leq \sum_{u_i \in Q'} v(u_i).$$

We show that the term in large brackets is 0. Thus, $V \leq \sum_{u_i \in Q} v(u_i)$, and so the Knapsack Decision problem has a solution. Each of the 1 terms in $f = <S, 1, 1, \cdots, 1>$ is realized by either a b_i or an f_i, for $1 \leq i \leq d-1$. The f_i terms contribute $\sum_{u_i \in Q'} s(u_i)$ to $f(0)$. Thus,

$S - \sum_{u_i \in Q'} s(u_i)$ copies of b_0 are needed. It follows that $l = (d-1) + S - \sum_{u_i \in Q'} s(u_i)$. Thus, a solution to KD$(Q, s, v, S, V)$ exists, such that $\sum_{u_i \in Q'} s(u_i) \le S$ and $\sum_{u_i \in Q'} v(u_i) \ge V$.

Q.E.D.

Since the Knapsack Decision problem is NP-complete, Lemmas 1 and 2 prove the main result.

IV. AN ALGORITHM FOR FINDING MINIMAL COST

In this section, we present an algorithm, MIN_COST, for solving the cost-table problem. Next, we analyze the time complexity of MIN_COST, showing how the number of steps depends on K, the size of the problem. We show that for smaller cost-tables, the complexity is exponential, while for larger cost-tables, the complexity is polynomial in the size of the problem.

A. MIN_COST

We present an algorithm, MIN_COST to find the minimal cost realization of a function f using the cost-table technique. Specifically, MIN_COST (F, f) finds a realization of f with minimum cost, $c_F(f)$, given any cost-table $F \subseteq U_{d,n}^r$ and any function $f \in U_{d,n}^r$. No other published algorithm is known. It is superior to the exhaustive search algorithm used in [7]. The algorithm for solving CD given in Section III is the nondeterministic version of a deterministic algorithm that searches exhaustively over all combinations of cost-table functions for a realization with a cost less than a given threshold. Searching for the *least cost* realization yields behavior that is identical to MIN_COST.

However, it is not necessary to search over all cost-table functions. Given two functions, f and e, let $e \lesssim f$ mean that, for every assignment A of values to the variables, $e(A) \le f(A)$. It follows that, unless $e \lesssim f$, e will never be used in a realization of f. Let $E = \{e \mid e \lesssim f\}$. $(E, \lesssim)$ is a partially ordered set, and the elements in E can be indexed such that, for all

$e_j, e_k \in E$, if $e_j \lessdot e_k$, then $j \le k$. Then, $e_0 = \mathbf{0}$ (the constant 0 function) and $e_{|E|-1} = f$. Let $I = (F \cap E) - BT$. I consists of all functions in cost-table F that are potentially in the minimal realization of f, excluding functions in BT. MIN_COST forms a sequence of cost-tables $BT = F_0 \subset F_1 \subset \cdots \subset F_{|I|}$, such that for $F_i - F_{i-1} = \{f_i\}$, where $f_i \in I$. MIN_COST begins by initializing $c_{F_0}(e_j)$ to $c_{BT}(e_j)$, for $0 \le j < |E|$. Then, for each cost-table F_i, where $1 \le i \le |I|$, $c_{F_i}(e_j)$ is computed for each $e_j \in E$. When MIN_COST reaches $F_{|I|}$, it has found a minimal cost realization of the given function f in cost-table F.

MIN_COST only checks for one use of f_i in the realization of any e_j. A complication arises if f_i is required more than once in the minimal realization of some function e_j. Consider the case where $e_k = f_i + f_i + e_r$, and $e_s = f_i + e_r$. Since $e_r \lessdot e_s \lessdot e_k$, the ordering over E requires that $r \le s \le k$. So $c_{F_k}(e_k)$ will be calculated using $c_{F_i}(f_i)$ and $c_{F_i}(e_s)$, but the cost of e_s will have already been updated using the functions f_i and e_r. Therefore, algorithm MIN_COST correctly computes the cost of functions which use multiple copies of cost-table functions.

B. THE TIME COMPLEXITY OF MIN_COST

1. The Time Complexity for a Single Function.

MIN_COST consists of two steps. First, the cost of each $e_j \in E$ using the basis cost-table is computed by summing over all functions in BT, requiring d^n operations or $\mathbf{O}(d^n\,|E|)$ operations for all e_j. Second, for each cost-table F_i, the new cost of each e_j is computed, requiring at most $\mathbf{O}(d^n\,|E|)$ operations per cost-table. Since there are $|I|$ cost-tables, the entire algorithm has time complexity $\mathbf{O}(d^n\,|I|\,|E|)$.

In [7], cost-tables for one-variable 4-valued functions were analyzed in order to study heuristics for finding minimal cost-tables. We can conclude that MIN_COST works well for cost-tables for such functions with sizes as small as 5 and as large as 256.

Algorithm MIN_COST

{ Compute costs of $e_i \in E$ (and thus f) using the basis cost-table }
$c_{F_0}(\mathbf{0}) := c(\mathbf{0})$
for $j := 1$ **to** $|E| - 1$ **do**

$$c_{F_0}(e_j) := \sum_{\substack{b \in BT \\ b(A)=1}} e_j(A)\, c(b) + \sum_{\substack{b \in BT \\ b(A)=1}} e_j(A)\, \sigma \; - \; \sigma$$

{where σ is the cost of adding two functions and $e_j(A)$ is the value (viewed as an integer) of e_j for the assignment of values A such that $b(A) = 1$. The left sum represents the costs of basis functions, while the right sum less σ represents the costs of adders.}

{ Compute costs of $e_i \in E$ (and thus f), using F_i, the next cost-table in the sequence }
for $i := 1$ **to** $|I|$ **do**
  begin { for f_i in I, where $\{f_i\} = F_i - F_{i-1}$ }.
  for $j := 0$ **to** $|E| - 1$ **do** {set the cost of a function e_j using F_i to the cost of e_j using F_{i-1} }
    $c_{F_i}(e_j) := c_{F_{i-1}}(e_j)$
  if $c(f_i) < c_{F_{i-1}}(f_i)$
   then
    begin { update the cost of e_j using F_i if it is less than the cost of e_j in F_{i-1} }
    $c_{F_i}(f_i) := c(f_i)$
    for $j := 0$ **to** $|E| - 1$ **do**
      if $f_i = e_j$ **then** $c_{F_i}(e_j) = \min\{c_{F_{i-1}}(e_j),\, c(f_i)\}$
      else if $f_i \leq e_j$
       then
        begin
        find h such that $h + f_i = e_j$
        $NEW\_COST := c_{F_i}(h) + c(f_i) + \sigma$
        $c_{F_i}(e_j) := \min\{c_{F_{i-1}}(e_j),\, NEW\_COST\}$
        end
    end { update the cost of e_i in F_i if it is less than the cost of e_j in F_{i-1} }
  end { for f_i in I, where $\{f_i\} = F_i - F_{i-1}$ }.

Table V: Formal description of MIN_COST, an algorithm for finding the minimal cost realization of a given function from a given cost-table.

2. The Time Complexity as a Function of Input Size

From the previous analysis, the time complexity of MIN_COST is polynomial in $|E|$. We now consider the relationship between $|E|$ and the size of the Cost-table Decision problem $K = d^n\ |F|$. Let F be a cost-table of size one larger than the basis cost-table; therefore $|F| = d^n + 2$. Let f, the function whose cost we wish to minimize, be the constant $r - 1$ function, so $E = U^r_{d,n}$, and $|I| = 1$. In this case, the time complexity of MIN_COST is $\mathbf{O}(d^n\, r^{d^n})$, while the size of the problem is $K = d^n\,(d^n + 2)$. Thus, MIN_COST's time complexity is $\mathbf{O}(\sqrt{K}\, r^{\sqrt{K}})$.

As the size of the cost-table $|F|$ increases, the time complexity of MIN_COST becomes polynomial in $|F|$. In the limit, $F = U^r_{d,n}$, and the time complexity of MIN_COST becomes $\mathbf{O}(d^n\ r^{d^n}\ r^{d^n})$, while the size of the problem is $K = d^n\ r^{d^n}$. Thus, MIN_COST's time complexity, $\mathbf{O}(K^2/d^n)$, is polynomial in the size of the problem, when the cost-table is sufficiently large (approaching $U^r_{d,n}$).

3. The Time Complexity for All Functions

In the process of finding a minimal cost of function f, MIN_COST finds a minimal cost realization for all functions $e_j \in E$. If f is chosen to be the constant $r-1$ function, then $e \lesssim f$ for all functions $e \in U^r_{d,n}$, so $E = U^r_{d,n}$. Using the previous analysis, a minimal cost realization of all functions can be found in $\mathbf{O}(d^n\ |F-BT|\ r^{d^n})$ time by MIN_COST. Thus, MIN_COST provides a more efficient alternative to exhaustive search algorithms, as demonstrated in analyzing various cost-tables [7].

V. CONCLUDING REMARKS

During the past fifteen years of research on cost-tables, there has been no computationally tractable algorithm for finding minimal cost realizations of given functions. We show that this problem is NP-complete. We also show that restricting the cost-tables to be minimal (the total cost of realizations by such cost-tables is minimal) produces no relief; the problem is still NP-complete. This result represents compelling evidence for the value of heuristic methods for cost-tables.

V. ACKNOWLEDGMENTS

The authors appreciate the comments by two referees which served to improve the manuscript. The research reported was supported by NATO Grant 423/84, by NSF Grant MIP-8706553, and an NPS Direct Funded Grant in cooperation with the Naval Research Laboratory.

REFERENCES

[1] M. H. Abd-El-Barr, Z. G. Vranesic, and S. C. Zaky, "Algorithmic synthesis of MVL functions for CCD implementations," *IEEE Trans. Comp.*, vol. C-40, no. 8, pp. 977-986, August 1991.

[2] J. F. Gimpel, "A method of producing a boolean function having an arbitrarily prescribed prime implicant table," *IEEE Trans. on Electron. Comput.*, vol. EC-14, no. 6, pp. 485-488, June 1965.

[3] R. M. Karp, "Reducibility among combinatorial problems," in R. E. Miller and J. W. Thatcher, *Complexity of Computer Computations*, Plenum Press, 1972, pp. 85-103.

[4] K. Keutzer and D. Richards, "Computational complexity of logic synthesis and optimization," *Proc. of the International Workshop on Logic Synthesis,* pp. 1-15, May 1989.

[5] H. G. Kerkhoff and M. L. Tervoert, "Multiple-valued logic charge coupled devices," *IEEE Trans. on Comp.*, vol. C-30, no. 9, pp. 644-652, Sept. 1981.

[6] H. G. Kerkhoff and H. A. J. Robroek, "The logic design of multiple-valued logic functions using charge-coupled devices," *Proc. of the 12th Inter. Symp. on Multiple-Valued Logic*, May 1982, pp. 35-44.

[7] K. A. Schueller and J. T. Butler, "On the design of cost-tables for realizing multiple-valued circuits," *IEEE Trans. Comp.*, vol. C-41, no. 2, pp. 178-189, Feb. 1992.